AVOCATIONAL COMPUTERISTS' PATHFINDER

FIRST PUBLISHED BY -
William J. Harmer
97 Ruskin Street
Ottawa, Ont. K1Y 4B3
CANADA

--------Amateur Computer User's Newsletter Tracks Techno-Trends Too-----#1-1993---
..Issue of Dec.31, 1993... ©1993 in the public domain, photocopy freely....8 pages.

Well, we almost missed the year without issuing a single newsletter of the traditional type coming from here, the 8 page reduced format. In fact, there has been some question as to whether it was time to stop putting these out at all, but the time for that has not come yet it appears. This year has brought the promise of real competition, with several microprocessor makers producing new IBM PC compatible microprocessors, and several companies producing alternatives to MS DOS as an operating system, DR DOS by Novell, being joined with a PC DOS from IBM for clones as well as true IBM make computers. In this newsletter, a little longer look is taken, to the technology of voice-input word processors and its effect on the job market. The training of non-technical computer users is examined on page 6, when the story of how the BASIC computer language was abandoned by most teachers as a way to approach increasing understanding of computers among their general students. It is looking like a case of the baby being thrown out with the bathwater, when teaching rudimentary computer programming to the average student was abandoned.

The rest of this newsletter is mainly a collection of attempts to try to find some of the things that the general student should learn about computers and try some ways of presenting them in an understandable way. Page 2, our MS DOS page, again tackles a topic of MS DOS command language, in search of ways of explaning this somewhat intricate way of interfacing with a computer user. On page 5, a story of the invention and evolution of disk drives is set out. The writer can admit by now, that writing about disk drives is becoming a bit of a personal obsession! An article commemorating the passing of B.F. Skinner, the famous psychologist, has been carefully saved from 1990, to place in this newsletter to try to draw some parallels between computer science and behaviourism. They certainly have influenced each other and a host of science fiction writers thinking about the limits of machine intelligence. The final article on page 8 considers some of the documented files and versions of both CP/M and MS DOS, for those who are into the practical parts of the history of those two operating systems, the most successful pioneering disk operating systems for microcomputers.

Last time, this writer offered some suggestions for improving databases, including some smart help features and layers and integrating screen grabbing with the database in some fancy way. Then yours truly actually tried to write a database program for GW BASIC, IBM PC compatible, and even with a lot of the work started, just barely got it working to use, but not well enough to distribute after a year and hundreds of hours of work. In the process, the world's slowest sorting routine was likely invented, and all of the 62K or so source code limit allowed by GWBASIC used up, and some valuable parts of the program even had to be deleted to get a working if simplified version into the available RAM at all. The end result, a comma delimited database for handling a small collection of classical records, is nice, but hardly fully featured, and not ready to compile yet either. Everything just seems to take so much longer than estimated with computer programming. At this point, putting in any extra features like those that this writer was recommending to others, seems a distant dream. But one thing discovered, is that there are not enough programs of all types for the old IBM PC's and their simple BASIC dialects, for either casual users or to use to teach computer programming or software internals. Even Kmart does not seem to carry programs for the old IBM PC types anymore, with the demand for hard disks, extended memory or faster processors, usually tucked into a corner of their packages. Judging form the amount of work involved writing the simplest program, it seems unlikely that a bunch of amateur programmers will fill the gap with shareware, something that we used to sort of count on in the early days of the IBM PC clones. It proves that it is a lot easier to write about computer problems than to try to produce industrial strength or commercial style programs to solve them!

....NOT SOLD ON NEWSSTANDS...CONTACT YOUR USER GROUP....A HOBBY SWAP PUBLICATION

Published occasionally, when the spirit moves this writer, since 1990. Send $2 if you write a letter for info to help cover costs. It's not a business or charity. It's not a government publication. It's just a personal hobby.................

Insight Into MS DOS Commands For Low Level Computer Users: The No-Command-At-All One

It is kind of funny, when computers were made so that humans could program them so they could do anything that the user wanted to program them to do, that they are rarely used that way anymore. Most IBM PC compatibles or clones are just a box into which the user shoves a disk containing someone else's programming, and whether run off that disk, or installed on a hard disk and run from there, the average computer is never programmed at all by the user or in that customized way that was originally intended. So computer programming became a specialty. Programmers were asked to write programs that would do anything that anybody wanted, but still not take up too much memory or be too complicated. Of course these were contradictory tradeoffs and so occasionally you get into a situation, using a prepared, off-the-shelf, standard package program, that we all use, where the program will do 99% of what you want, but to get it to go the rest of the way, seems impossible. Luckily there is a way aout, a trap-door left by the designers of MS DOS computer systems, and if you find your way in there, you can do some low level file manipulation, interfacing with printers that the program designers did not think of, or other programs, and so on. This is low level DOS command work, and it offers many possible tricks for the case where it is late at night, the work has to be done, and so it seems that your application program, has allowed you to paint yourself into a corner. I mean, it seems there is no way to get from were you are, using the application program, a word processor, or whatever, to where you want the work to finish.

Well, how do you make a computer of today obey your specialized wishes? How do you seize personal control of its programming power? You can do it several ways, without learning any of the old programming languages like BASIC, COBOL or FORTRAN, not to mention that mathematician's delight, the all-numbers machine language. We assume that if you read this far, you are talking about an IBM PC compatible or clone using MS DOS or PC DOS rather than Windows. For such a computer, it is still possible to find certain opennings for inserting your own instructions in MS DOS command language. It is hard to generalize with so many computer systems out there that are slightly different in the way they are set up, but many of the older style systems will start with a blank screen, after the cursor, or position where you can type in anything, and the prompt will be a letter of the alphabet followed by, at least a colon, such as A: or C: and these show that MS DOS commands can be entered. There may also be other opportunities for entering MS DOS commands inside a program, such as the so-called "SHELL" feature of some wordprocessors, etc. or running a series of MS DOS commands as a small program, by putting a batch file into the system (in an AUTOEXEC.BAT file on the system disk, or system part of the DOS subdirectory on a hard disk). But no matter where you find an opportunity, the question that most ask themselves, is that of how to understand DOS commands.

DOS commands are made to answer a number of questions. "Where?" is one of them, and since DOS looks for a something to do, and something to do it with, and must know where to look. To find these things, you would think that DOS commands should be divided into different parts, parts answering the questions "where?," "what?," "why?" and so on. What to take as input and what to do with output. But, in order to save on typing, and the engineers and scientists for which this type of command language was first developed did not want to type very much, the language turned into a sort of shortened version, in which as little was said (into the computer) as possible.

The least that can be said to a computer when telling it what to do, is of course nothing at all. You could just tell the computer what to do it with, and not tell it what to do. For that reason, one of the first commands in MS DOS to learn is the command that does not tell the computer what to do, since it knows that at the very mention of certain kinds of files on disk, to go ahead and run them like running a program. It may be confusing to avoid using a command. (In the BASIC computer language, a command is used to do this, RUN, followed by the name of the file to be run, assuming that file is a basic program). But that is how MS DOS does it. Use no command, but just the file name, and if that file name is a program that can be just run by MS DOS, with no further action, it will be run. These programs are seen on disk, as those ending in the extension, .EXE or .COM and, a special case, batches of MS DOS commands that run like a small program, those files ending in .BAT. So to run a program, when you see it on disk, the command is, no command at all, but just use of the name of the disk file. Like "Rover", when you call a dog, or a file on disk, just using its name will cause it to come up running. That is, if it is where you are calling. Just like calling for Rover out the front, when he is out the back, and not listening, your program may not come running if it is somewhere else. But that is another story for another day.

The Next Revolution In Office Computing And How To Be Ready For It

It is getting hard to remember when the clicketty clack of electric typewriters was the only way to get text out of an office or around in the form of inter-office memos. Today, a lot of text never touches paper, and indeed in some offices these days, hit by economic pressures, there is a move to reduce paper use as much as possible. Today's office is built on doing as much as possible, typing (or keyboarding as it is now referred to), only once. This is starting to result in more of the work that typists used to do, perhaps not even needing to know spelling, being done on a level of intelligence of computers that can check spelling and catch some grammar mistakes too. This saves human checks for the editing of documents on higher and more subtle levels. If scanners and voice-input word-processors are just around the corner, the bottom level of this process, the manual keyboarding, may also be cut to a fraction of the work it now provides. That means that although the reluctance of businesses to spend on new hardware and the lack of need for it, with business generally slow, may delay it, the largely female staff doing this work now should be looking to learn new levels of skills that will enable them to survive the shake-up and redundancies that will inevitably come. Many will try to move into accounting using spreadsheets, some into data work as well as the higher level editing and administration. It would also be a good idea to get into the theory of use of computers, low level and high level now since those most affected will have little time when this transition comes. It is time right now to start understanding some of the finer points of computers — as well as English and other business subjects.

That raises an interesting point in that while computer user group membership may be valuable in this upgrading, as many women are finding, there seems to be a gap in the line-up of educational materials available for this upgrading. In order to search for ways of filling this gap, of ways to explain low level disk file and data structure and operations in terms for someone working in an office and not particularly technical or electronic oriented, this writer has been looking into the sort of concepts and material to use. Why not pick common things in an office to use in these explanations, rather than try to build up a vocabulary of technical terms like bit and byte, track, cylinder, etc. from scratch? Maybe one of my early attempts at writing this sort of low level explanation of today's computer systems, could be presented in a future newsletter issue. To ready it for a real book the text would have to be spread more thinly among more pages and alternated with more drawings by way of illustration, of course, but we will see.

For writers, managers and computer users of a more casual level, this looks from here to be the sort of material that will be needed more and more in the future. It is hard to have confidence in this material being done well in the future when major books by major, well reputed writing groups and publishers, will plunk down in the middle of an explanation of low level data structure, a few statements of things that just are not so. That means if experts or hacking enthusiasts are not going to be writing this a lot of what will be written will be worse than uselsss; it will be misleading, and lead to discouraging those who should be studying this sort of material. It will also be untestable, if simply stated in prose in a book. So what we need is both explanations in text and ways of testing them. Since low level hacking is too hard for the average student, that means putting data handling exercises into some sort of computer program routines, like BASIC.

....

The Revolution Begins - Data By Wire Comes Before Data By Voice

Technology is accepted or rejected on economic grounds and today, we might well ask, why we should worry about new types of automation sweeping into the work world, when increases of automation are driven by the economics of labour scarcity, something that the current recessionary times are not characterized by? One answer is that companies and governments will continue to make economies in labour use that save money, and are easy enough to plan. What this means is that existing technologies are likely to be pushed further, even if newer ones that would require a whole new range of equipment are not. Compatibility is the watchword for economy now and those who can quickly learn a new piece of software for an old type of computer like the IBM PC compatible. will be employees worth their weight in gold. They have what in computers is called 100% compatibility.

Voice-input word processors that merely need dictating to, may be ten or twelve years away from replacing very many desktop, keyboard-based computer applications, and of course will never replace them all, although transforming them is likely. The big revolution that might eliminate some more jobs and change the average job is the method of computer input that goes by wire rather than by hand, networking in other words. That means that the concepts of network data sharing, filtering and translating, as well as the protocols of transmission, would be the next concepts to look at for low-level computer concept courses. Data structure and the way data files are broken into blocks, similar to disk files being broken into sectors for saving on disk, would be a good concept to employ to judge a book on, as to whether it really gets down to the low level or skips across the top.

B.F. Skinner And The Computer Hacker

With the passing of B.F. Skinner, the American psychologist known for behavioursim, it is probably a good time to mention his relevance to the spirit of our time as represented by the way working with microcomputers has affected the way we think about thinking. Prof. B.F. Skinner seemed to want to take 'thinking' as an explanation of why people do things, right out of psychology. He wanted a psychology that in explaning a person's reason for doing anything, would call only on descriptions of his behaviour as observed by an external, impartial observer. He did not trust the often vivid descriptions of the ideas going through people's heads when they did this or that, as being useful in explaning their actions. He believed that these ideas were an appearance of things being decided rather than the real seat of decision, a by-product of the real determining factors. He hoped that the real determining factors were much deeper and more describable by laws similar to those of physics, and his unique contribution to the psychology of behaviourism was to propose a deep-seated type of conditioning, operant conditioning, a sort of association that determines behaviour ultimately regardless of the ideas going through our minds on the surface, while it is happening. Our body goes along the tracks of this operant conditioning and our ideas no more determine its course than the screams of a roller-coaster rider. Our ideas and vivid sensations just go along for the ride of life and like a roller coaster rider, have no hands on the control levers of our life, psychological(life)or otherwise. (In a strange twist some Freudians who also believe in deep unconscious factors prevailing over surface, rationalisms, also bought into behaviourism, at least in the U.S.)

In a world of the 1950's, the heyday of this type of psychology, control mechanisms seen in the machinery that was the model for this kind of mechanistic, materialistic thinking, the control mechanisms of say the guns and missles of the day (and B.F. Skinner worked on pidgeon guided missiles), were simplistic, analogue computers at best with only a few factors serving as input data for their 'computing'. In today's world, where even a simple, orphan home computer can be programmed to take into account hundreds of factors and run them through complicated equations of an arithmetic or Boolean kind in long complicated sequences of instructions through almost infinite loops, it is a little harder to be convinced of the existence of simple laws, unrelated to hard thinking and logical processes, as the way that human beings solve problems in the real world. We do not solve these problems by using long equations or Boolean expressions, juggled in our minds like absent-minded mathematics professors, splitting our attention between the calculations in our head and the world whizzing by in front of our noses. But it would seem to most of us who work or play at computer programming that the human mind in a problem solving situation will revert to some sort of rough and ready, abreviated or trial and error approach to solving the problem using logic of a less formal type than the straight Boolean of computers. Nonetheless it is based on the same logical features of the world and events and can just as easily be simulated simplisticly by a computer set up with a program to run the problem by solving a series of Boolean or arithmetic equations. The computer is much more determined by the formal details of how the equations are set up and happily will do what your tell it, going by form only, rather than what you meant to tell it to do. But is would seem that both the computer and the human being have come to grips with the logical relations between events in the world, use them to make decisions, and use the decisions to choose between alternative actions. Leaving the logical, mental decision making process out of psychology would seem to our generation, the first to get into the nitty-gritty of computer use at the lowest, most nuts and bolts level of understanding, to be an oversimplification of the laws of behaviour either human or that of the new computer controlled robots. On the passing of B.F. Skinner, this is his epitaph, that his life saw changes in science and machines so radical that never again would a human generation compare a person to an unthinking piece of steel or a biological function like a reflex.

But again, B.F. Skinner simply put a once new twist on a school of psychology that goes back very far indeed (beyond Henry Regius, contemporary of DesCartes and Spinoza). That school would replace the often frivolous attribution of our life's destiny to decisions taken on 'feeling a whim' oblivious to the complicated and deep-seated forces within our being and in the external world that pull our strings more than we care admit. Behaviourism in some brands already has the seeds of new complexities, organismic variables, and possibly context analysis. It is not dead.

From Disk Packs To Today's Hard Disks And Floppies

The computer disk system was invented at IBM, first in the form of the hard disk, and it was invented to solve the problem that was reducing the efficiency of the magnetic storage device before that, the magnetic tape, that to find a piece of data or a program to run on a long tape, the tape drive would have to spend a lot of time in fast formard and reverse modes, seeking out the spots where the programs or data records were kept. Access to a specific item of data on tape varied in the time it took, one far away from where the tape drive was currently reading, would take a lot longer to get to by fast winding, than one right close to where the read and write heads were opposite. Time to access thus would vary so much from one data read or write to another, that the device of the designers wanting to speed up this access bottleneck, was to place the data close together, for serial access, reading each desired data group one after the other in the serial order they occurred on the tape or having to waste massive amounts of precious computer time with nothing happening but the tape drive motors spinning the tape. The quest then was for a magnetic storage device that could be used for true random access, that it would take the same short time to get to a chunk of data anywhere on the magnetic media, the average or typical case, the data placed at any location chosen randomly, which is the way mathematicans build a model to analyse such data access. With a fixed and short access time, operating system design could be much neater. Since access time was more predictable (depending now on only how much data was to be read in bytes, not where it was on a tape) i/o routines could then avoid time wasting due to synchronizing data read/write operation slots to an unpredictable data seek or write location seek time. The hard disk unit with its removable disk pack, kept under a plastic, lift-up hood for dust protection was the result.

The next step in disk development came with the increases in density of recording bits on each millimeter of data track. It got so dense that dandruff, dust and even cigarette smoke would cause losses of crucial bits of data or program crashes, if it was a program stored on disk, so hermetically sealing the disk in the unit, made the hard disk units into fixed disk units. To get data into a computer in quantity, since hard disk packs could no longer be shuffled into and out of the disk unit from the data and program library, a new form of specialized disk system was developed at IBM, the floppy disk. First ones, 8 inches in size, were clumsy, made of the same thin polyester plastic used at the time in better quality audio tape and the disk placed in a flexible cardboard jacket, never to be removed, Such disks, feeling floppy in the hands of computer operators, soon earned the name, floppy disk. This form of disk was a real step up for home computers, early CP/M business microcomputers adopting the IBM standard format 8 inch floppy disk, for its two disk drives, the A: or program disk and the B: or data disk, being quite an adequate substitute for the more expensive hard disks used in the big computer installations, and of course about $9000 cheaper! The old 8 inch floppy disk seems primitive today, oversized and of limited capacity, but it soon led to miniaturized versions which were much more efficient.

The first version, called at the time the diskette or minifloppy disk, really became a popular standard. It was 5¼ inch, with a floppy cardboard cover and after improvements making it double-density over single-density, double-sided over single-sided for data use, and all 40 tracks on both sides (total 80 tracks) being used, compared to earlier drives that used about 35 tracks per side
In the most widely used format, IBM PC or MS DOS, it had 360Kbytes of capacity. All hobbyists using IBM PC clones have made this the most popular disk format and disk size in use, shareware and commercial programs alike being distributed on this size and format of disk more than any other.

However progress did not stop there. By using a narrower recording track and a smaller read/write head gap, 5 ¼ inch disks using an improved, finer coating, (5¼ inch HD disks) boosted data capacity to 1.2 megabytes (1200 Kbytes) in the newer 80286-80486 IBM PC/AT compatible clones. A quad drive with 5¼ inch disks did not really catch on, and used a special drive too, like HD disk systems did, so that a normal DSDD disk drive will neither read nor write in HD format or quad format. Even with an HD floppy drive some special software is used to have the higher capacity drive systems emulate DSDD format for example, and its 360K limit.

After experimentation with 3¼ inch drives, adopted by some European Sinclair computers and Smith Corona electronic word processing typewriters, came the 3½ in. floppy disk. Now in a hard shell rather than a flexible cardbaord sleeve, and so no longer floppy in one's hands, although the inner polyester disk remaining as flexible and floppy in nature, it is the new standard for IBM PS/2, IBM clone laptops and the 68000 computers like the Apple Macintosh, Atari ST's, and the Commodore Amigas. It comes in single, double sided, DD and HD flavours too and is seen more often, gradually becoming dominant over the 5¼ inch ones.

Teaching Computers With The BASIC Language-Rise, Fall, Repudiation And Repression

The BASIC computer language launched home computers. "Is it a useful language or was it a disasterous invention?", is a question still debated today, one that still arouses bad temper among more than a few. To admit my biases, let us just say that: I seem to be following one of the less desirable patterns of conversion, if the commonest, moving from being a once enthusiastic BASIC programmer as a hobby to losing interest in computer work itself. My interest now is the history of computers more than anything. This story of BASIC is a major part of it.

It is often forgotten that the BASIC programming language was used for 20 years in teaching computer programming and it reached many not usually capable of being induced to study anything very technical. It was designed by members of the Mathematics Department of the USA's Dartmouth University. That was also where the mathematics textbook "Finite Mathematics" came from, which was an attempt to teach social science students, also a traditionally math avoiding group of students, such fairly advanced topics as mathematical logic and matrix and vector algebra, giving applications to various social sciences as problems for homework. The invention of BASIC proved wildly popular in winning the enthusiasm of many students in US educational institutions, as witnessed by the books edited by David Ahl, that collected BASIC games that students produced in their spare time, using the BASIC programming languages and the BASIC available at colleges and universities on the paper/typewriter clunkamatic terminals connected by time sharing systems to the university computers of the time. The courses in computer programming spread to the community college business students and even high schools, accelerated further by the availability of the early generation of home computers in which BASIC was the built-in language and operating system command language as well, in the 1980's. In fact from the time Microsoft's William Gates produced the first BASIC interpreter for microcomputers in 1975 for the 8080-based microcomputers being home-made in the USA, BASIC and microcomputers have proven a popular coupling. BASIC is still the language that comes as standard for IBM PC compatible microcomputers and MS DOS operating systems.

When BASIC was chosen to be built into microcomputers sellers succeeded in overcoming the lack-of-software-problem that accompanies any new computer hardware release. How? Hundreds if not thousands of home computer hobbyists rushed into filling the gap in software for a dozen popular models with incompatible BASIC dialects, their programs sold mailorder, published in magazines and books and distributed among computer hobby user groups at meetings, (when they had been placed in the public domain by their designers). Attempts to sell these same microcomputers with FORTH as the built-in language were unsuccessful. If the IBM way of using machine language (as on the IBM 1620) or a highly numeric and cryptic operating system interface like JCL, had been followed with microcomputers it is doubtful if very many would have been sold in the 1980's. In the user groups BASIC was particularly easy as a computer language to learn without actually taking a rigourous or any course at school. Many people started their careers in computing by learning programming in BASIC on their own, without the spending of a cent by any government educational institution, using these home computers.

Having stated the case for BASIC, including some aspects that are forgotten today by the keener critics of BASIC, it is time to look at the history of the other view of BASIC which many in the computer professions, especially today's teachers at post-secondary level espouse now. We will not dwell on the debate but its history. It starts in 1965. It was discovered that you could write a whole computer program without using GO-TO (in ALGOL), and this soon was hailed as the only way to program, whole languages being invented which did not have the GOTO instruction, and some like Pascal updated. Then Prof. Yourdon was teaching classes on better computer programming, wandering all over the world and finding and correcting the bad habits of programmers that were making their work inefficient. IBM, also concerned with inefficiency and the shortage of software that was slowing the potential sales of its computers tried, in the superprogrammer experiment, some of these new methods called modular and structured programming, so that one good programmer and some assistants less skilled could do the work normally requiring a team of programmers. One of the first and likely most fervent converts to these new methods were the teachers who put aside their BASIC programming courses and started saying such things as,'A student taught programming in BASIC will, never be a good programmer, having acquired bad habits of fixing programs as they go along and not planning them properly from the start using modular and suchlike methods'. Having taught a generation of such students precisely that way, there would seem to be an element of guilt mixed up in and emotionalizing this debate against BASIC. BASIC relies on the GOTO. It is hard to optimize its programs for speed and frugal use of memory if applied rigidly to BASIC programming as can be seen by looking at BASIC programs of the home computer era.

(CONT. P.7→)

Continued From Page 6, Teaching Computers With BASIC (gen.tail)

At this point, the teaching of computers took a fork in the road. One path led computer nerds and technical experts to go with the post-secondary teachers who promised never to teach them anything not in a structured language, if it could at all be avoided, even with machine language programming. All the rest floundered, and even our teachers seemed at a genuine loss as to what to teach or how to go about it. They tried teaching programming in the schools, as if nothing had happened, but students thinking that computer courses meant just playing with video the way they did with their video games, soon got bored with attempting and having only limited success in learning programming. Teachers started looking around for alternatives, and other than teaching word processing program use, spreadsheet and database program use, but there was disatisfaction with knowing what of lasting use was being taught under the heading of what it came to be called, computer literacy. The criticism coming from within the profession, is that such courses teach only the shallowest details of computer use, the sort of thing that most workers have learned without taking any special course.

The fact is, the demand for computer educated employees focusses on ones that know how to do a little troubleshooting and fast learning of new equipment and program versions. It means that those educated in the word processing programs of today, may not be any better prepared for the future than those who ten years ago were learning key punch operating, (with the old punched card machines not seen anymore today) are prepared for using a modern microcomputer. It would seem that it is possible to teach at least some of the non-technical students more fundamental things than using programs or computers that may be obsolete in ten years. And part of these fundamentals would seem to be programming, not to teach students to be programmers as a job, but to understand what happens inside the software of anything computerized. That sort of learning is likely to be the most useful in developing skills at troubleshooting problems and learning how to use new computer based hardware and software packages more quickly. BASIC programming itself can be used this way, as it was before, to teach software concepts and their interaction with hardware, from PEEK's to POKEs and beyond. It is hard to see any more promising approach to teach computerization and its concepts. Perhaps it was a mistake to abandon something that could have been taken a lot further towards perfection. What the situation looks like now is an abandoning of the whole idea of teaching the average student very much about computers and their internals, software or hardware. A clear win for elites and those who teach them over the rest of us. A turf war that has destroyed the initially receptive attitude of the average student had towards learning about computerization. What we may get as a future is merely a return to the technopeasant masses voting or acting blindly rather than as a force to humanize computerization and its consequences. What this may do to democracy is one thing. What it will do to economic competitiveness in our future may be much less subtle and much less tolerable for all in our society.

Views And Other Shorts - Education By BASIC, One Short Story

While teaching computer understanding with BASIC seems to have fallen out of favour in the classroom, learning by oneself using BASIC is still a viable alternative, with some computer user groups still sharing programs. This writer is experimenting with the way word processing programs work internally, these features being duplicated in GWBASIC on the IBM PC compatible. The result is a much greater appreciation of the nuances of word processor program use and behaviour. One project is the duplication of the global text processing features of some of the old, mainframe, line-editors in BASIC on an IBM PC clone. Those early text processors intended for the use of programmers, had both an instruction interpreter (although the commands of its command language were often only a letter or two long, plus some number(s) for line numbers, or strings) and the actual routines to delete, copy or insert material, from the text in RAM. In comparison in fact, the primitive MS DOS line editor EDLIN looks quite sophisticated. But working with even such old software features, is interesting and does also help retrace some of the history of computer software, in this csse from the 1970's mainframe computer world. So BASIC is still useful for something educational!

System Disks - MS DOS And CP/M

MS DOS has been upgraded numerous times, with 5 full version numbers having been released. Since MS DOS version numbers beginning with 1 had some bugs and awkward wrinkles, MS DOS version 2, (and 2.11 is the commonest) has been for years the standard normal minimum for someone running an MS DOS computer with a floppy disk and version 3.3 if the computer is equipped with a hard disk, since that version has certain improved hard disk handling features.

However, MS DOS version 1.25 may still be encountered, as it was bundled with some early computers, but it is further off the standard even for floppy disk equipped computers, since it used 320K DSDD 5 1/4 inch floppy disks as the standard double-sided format (and it also used single-sided format) rather than the 360K that most IBM compatible programs, public domain, shareware and commercial, come on even today. But the use of 3 1/2 inch 720K DSDD disks and high density disks is starting to rival use of the old 360K format, so that later IBM PC compatible computers, some only equipped with 3 1/2 inch floppy drive(s) introduces still another disk compatibility problem among tbe IBM PC compatibles.

The fact that some of the earlier MS DOS computers were not 100% perfect clones of the IBM PC, also makes for problems, the Sanyo 555 and DEC Rainbow being most different, the latter described as MS DOS compatible but not IBM PC compatible. Even the early Tandy 1000, 2000 etc. had some differences such as some different keyboard scan codes (for which fixes like the GreyKey program exist). For the early Tandy 1000's these actually make little trouble for users especially when compared to the wider variations in early CP/M computer hardware. Nonetheless, the user should be aware of the possibility of encountering such incompatibilities in MS DOS.

While, other than the MS DOS version 1.25 which uses 320K DSDD disks, all MS DOS, IBM PC compatibles (and most AT and fancier compatibles), can use a common disk format, the 360K 5 1/4 DSDD, in the CP/M world there is no common 5 1/4 inch floppy disk standard. As a result, commercial and shareware programs have been made to convert between disk formats, the Xenocopy being merely one of the commercial ones. Programs to read CP/M disks from MS DOS computers are less numerous than programs to at least read and write text files to MS DOS 360K floppy disks using a CP/M computer, but both are seen in public domain/shareware disk lists and specialty program sellers price lists.

CP/M has many more variations between too, versions, CP/M 2.2 and the more advanced CP/M Plus being the most common and a number of CP/M clones are also out there and occasionally seen. Some of these various types of CP/M compatible operating systems are not compatible at all. In fact, the problem of compatibility is more of a headache in the world of CP/M than for MS DOS computers and the MS DOS operating system. Some are relics of various attempts to improve the sales of CP/M including MP/M a multitasking version, and some are specific to computers with certain microprocessors. Some versions need a Z-80 microprocessor while others will run on computers with the older 8080 (and 8085) that CP/M started out as supporting. Since the Z-80 microprocessor has all the machine language instructions that the 8080 has, as well as some extra machine language instructions that will do certain things faster. The extra instructions that the Z-80 handles and the 8080 does not use, speed up certain operations and also save some memory. You may even come across a a version of CP/M for the 8088/8086 microprocessor, called CP/M 86 meant to be used only on IBM PC compatible computers in an unsuccessful attempt to beat MS DOS in that market. (An IBM PC version of CP/M Plus called DOS Plus is also seen on some computers, mainly those sold in North America a few years ago by Amstrad.)

Even a 6502 microprocessor version of CP/M was written, and promoted for use in computers like the early Ohio Scientific ones, although most 6502 type computers that run CP/M like the Commodore 128 and Apple II (when the latter is equipped with an expansion board to run CP/M), use a Z-80 microprocessor in addition to their standard 6502-family microprocessor so that they can run a normal version of CP/M and anything else in the 8080 machine code, such as programs compiled under standard versions of CP/M using standard 8080 object code output compilers.

The most popular CP/M version is version 2.2 and that is as close to being a minimal standard as CP/M ever got, (which is not very close). The Commodore 128 uses CP/M Plus 3.0, CP/M Plus being an optimized version of CP/M developed later to make full use of the extra Z-80 microprocessor machine language features that the 8080 did not have. Of course earlier and later versions of 8080 compatible CP/M than 2.2 exist, and cause some compatibility problems for the unwary when encountered. The more serious problems are however caused by the odder variants, and disk format incompatibility. Some IBM PC compatible emulators rely on the computer being equipped with a V-20 (or V-30) microprocessor which has an emulation mode to emulate the 8080 but not the Z-80 instructions, a V-25 planned to do that never having been released.